Chapitre 2

Une démarche d'une conception logicielle

L'objectif de ce chapitre est :
- de présenter la démarche de conception utilisée dans ce cours,
- quelques rappels sur UML,
- diagrammes de communication et de classes

1 Notre démarche de conception

La démarche de CONCEPTION que nous allons utiliser est basée sur le type d'architecture dite "Architecture à Base de Composants" (voir le cours précédent).

En général, les grandes étapes de la réalisation d'un logiciel sont :

- Analyser le besoin
- Réaliser la spécification du logiciel
- Réaliser l'architecture du logiciel :
    - Réaliser la Configuration Architecturale et la Dynamique de l'Architecture
    - **Réaliser l'Architecture Technique**
    - Réaliser l'Architecture Physique
- **Réaliser la conception de chacun des Composants techniques**
- Réaliser le développement de tous les composants
- Valider le logiciel

Dans le cadre du cours NSY 102, à partir de l'énoncé d'un exercice ou d'un examen, il vous sera demandé de faire :

- une Architecture Techniques à travers la réalisation d'un **diagramme de communication**
- la Conception de chacun des Composants techniques à travers la réalisation **de diagrammes de classe**.

On part d'un cahier des charges (énoncé) d'un Système d'Information (SI) qui décrit le besoin et la spécification du SI à concevoir et l'identification des COMPOSANTS techniques utilisés pour remplir le besoin.

L'objectif est alors de faire une conception statique et dynamique du SI en concevant le diagramme de communication. Puis de faire la description détaillée des Classes (UML) de certains des COMPOSANTS.

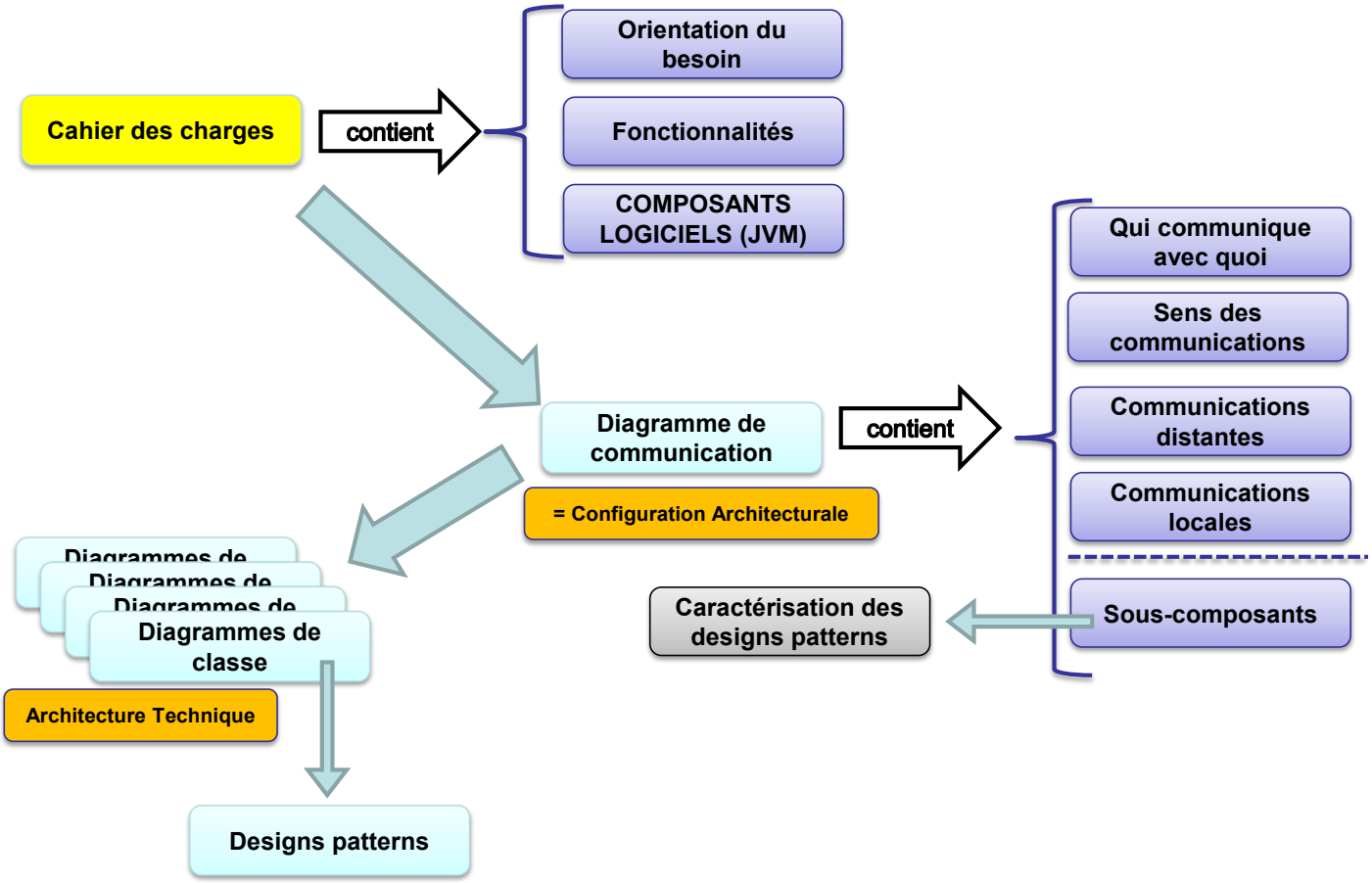
Notre démarche de CONCEPTION se fait donc en 3 étapes :

1/ Lire le cahier des charges du SI (énoncé)

2/ Réaliser le diagramme UML de **Communication** de l'intégralité du SI

3/ Réaliser le diagramme UML de **Classes** de certains des COMPOSANTS techniques en mettant en évidence les **Designs Patterns**.

C'est cette démarche qu'il faudra dérouler le jour de l'examen sur une problématique donnée d'un SI :


Cahier des charges
contient
Orientation du besoin
Fonctionnalités
COMPOSANTS LOGICIELS (JVM)
Diagramme de communication
contient
Qui communique avec quoi
Sens des communications
Communications distantes
Communications locales
Sous-composants
= Configuration Architecturale
Caractérisation des designs patterns
Diagrammes de classe
Architecture Technique
Designs patterns

2 Le COMPOSANT

Le COMPOSANT existe sous 2 formes :
- le composant "fonctionnel" qui peut être de très haut niveau, et qui ne présume pas de sa nature ou de la technologie utilisée pour son développement
- le composant technique et/ou matériel (physique) qui est fonction de la technologie utilisée pour implémenter le composant et des choix de déploiement sur les machines à sa disposition.

Dans notre démarche de CONCEPTION, le COMPOSANT est un composant technique et matériel afin de cibler la solution car notre objectif est de **décrire** les classes de ce composant (faire un diagramme de classes UML).

Ainsi ce COMPOSANT devient un COMPOSANT LOGICIEL qui sera, dans notre démarche, une application (ou programme) qui exprime des interfaces DISTANTES afin de COMMUNIQUER avec les autres composants.

Exprimer dans la technologie JAVA, on fait le choix suivant :

COMPOSANT LOGICIEL = JVM
Interface DISTANTE = RMI[1] (UnicastRemoteObject + Remote)

[1] Remote Method Invocation

3 Le diagramme UML de Communication

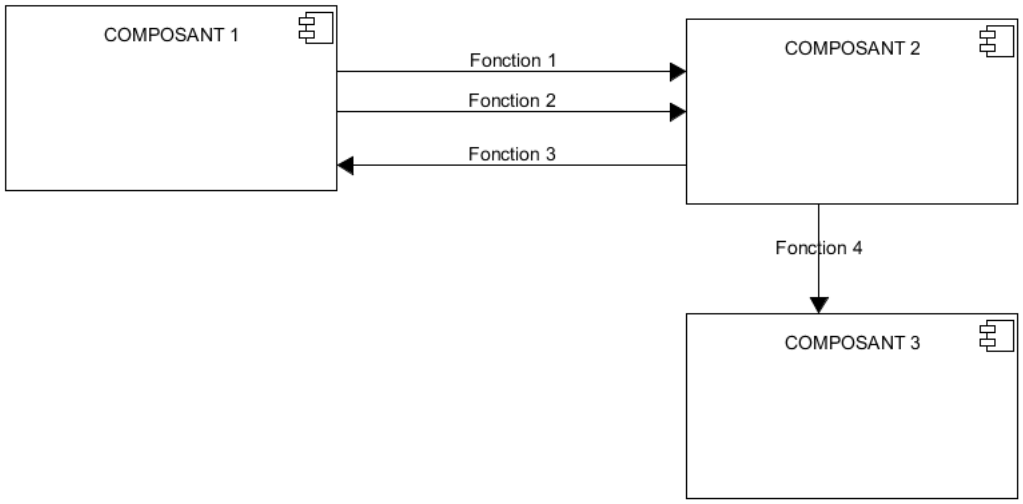
3.1.1 Définitions

Le diagramme de Communication[2] appartient à la famille des diagrammes UML d'Interaction (au quelle appartient aussi le diagramme de Séquence).

La définition qui suit est plus restrictive que celle standardisée par UML afin que cela corresponde à notre besoin.

Elle consiste en un **graphe** dont les **nœuds** sont les COMPOSANTS LOGICIELS, et les arcs (numérotés selon la chronologie d'un use-case si nécessaire[3]) sont les **échanges** entre ces objets. Ces arcs sont appelés des CONNECTEURS.

Un CONNECTEUR est toujours **orienté** dans le sens **appelant→appelé** car cela correspond à l'appel d'un traitement distant entre deux composants[4]. Il est important de noter que s'il y a un retour d'information celui-ci n'est pas représenté par un arc orienté dans l'autre sens. Cette valeur de retour est implicite. Elle peut être précisée dans le nom du CONNECTEUR ou dans un texte annexe qui commente le schéma.



[2] Les diagrammes de collaboration ont été remplacés en UML 2.x par les diagrammes de communication.
[3] La chronologie correspond souvent à un Cas d'Utilisation et anticipe (ou est le résultat) d'un diagramme de séquence dont les Acteurs sont les COMPOSANTS.
[4] ou l'envoi d'un message ou èvenements

3.1.2 Décomposition

Afin de gagner en précision et en justification, il est souhaitable de décrire aussi les SOUS-COMPOSANTS d'un COMPOSANT LOGICIEL.

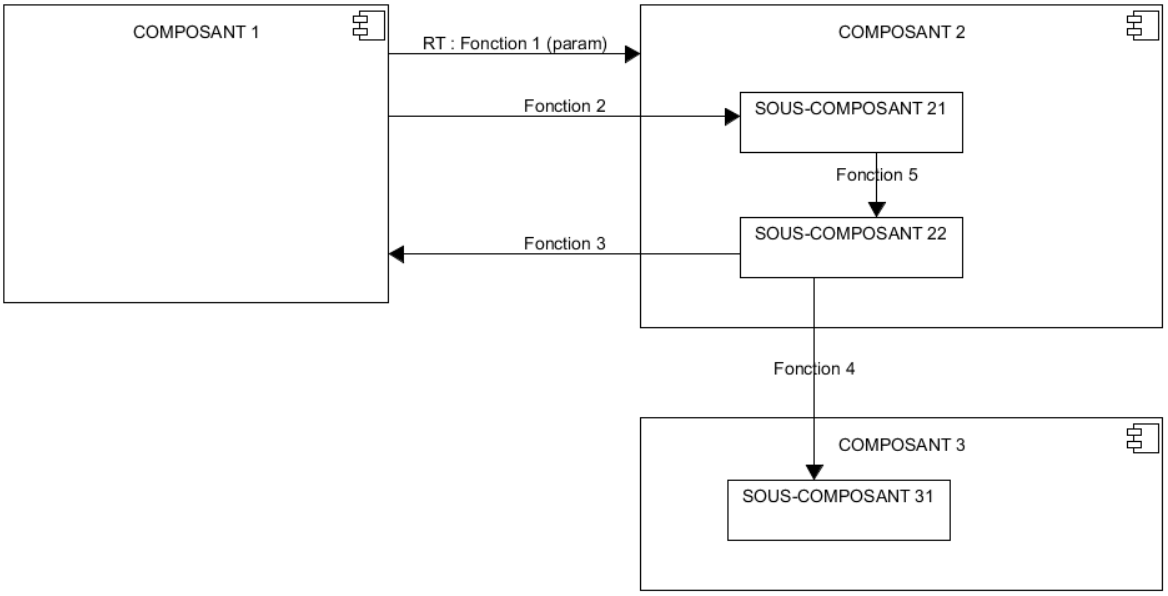
D'autant que ces SOUS-COMPOSANTS sont très souvent des **Designs Patterns** à part entière :

- Factory
- Builder
- Observer
- Iterator
- Strategy
- Model / Vue / Controlleur
- Proxy et Adaptateur de communication
- Objet Distant (RMI)
- Provider (ou MOM)

Cela permet d'anticiper l'étape suivante des diagrammes de classe

Les sous-composants d'un même composant communiquent entre eux à travers des connecteurs internes correspondant à un appel local.

La communication entre deux sous-composants appartenant à deux composants logiciels est une communication distante.



La Fonction 1 est entièrement décrite : type de retour (RT), nom de la fonction (Fonction 1), le(s) paramètre(s) (param).

Le COMPOSANT 1 demande au COMPOSANT 2 de réaliser la Fonction 1. On ne sait pas plus qui dans le COMPOSANT 2 réalise cette fonction. Cette fonction correspond à un appel distant entre les deux composants.

Cela signifie que le COMPOSANT 2 contient un élément technique qui est en attente de réception d'une requête. Nous verrons en détail plus loin que cette requête est l'appel d'une **méthode distante** décrit dans un Desing Pattern particulier.

On détaille la structure du COMPOSANT 2 avec 2 sous-composants.

On sait quel sous-composant réalise la Fonction 2 qui est aussi l'appel d'une méthode distante.

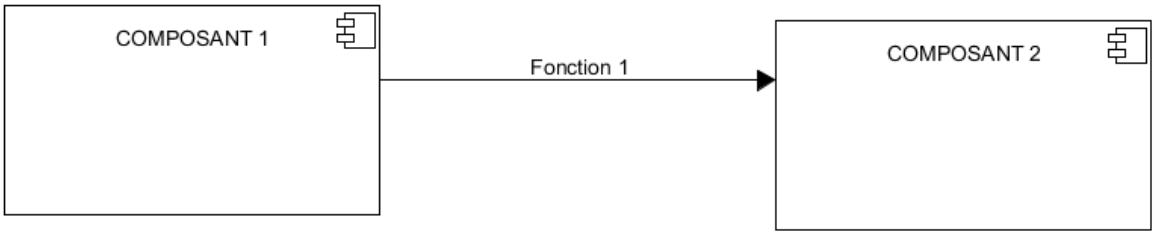
On peut décrire les échanges internes entre les sous-composants.

La Fonction 5 est un simple appel de méthode entre deux classes (appel local).

La Fonction 4 est de même nature que les Fonctions 1, 2 et 3 mais entre deux sous-composants et correspond à un appel distant.

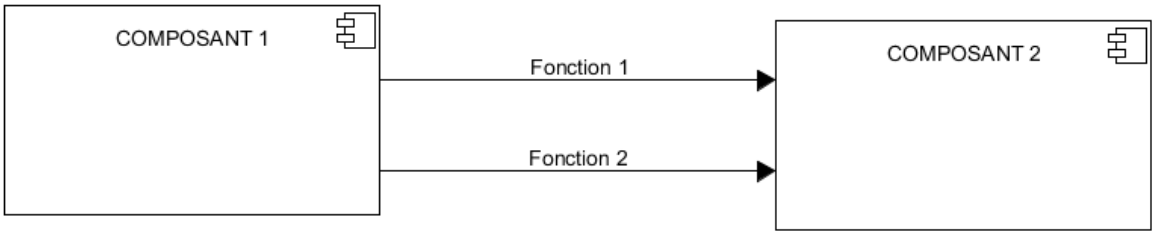
3.1.3 Communication

L'arc qui relie deux composants est orienté et les deux composants sont distants l'un de l'autre.



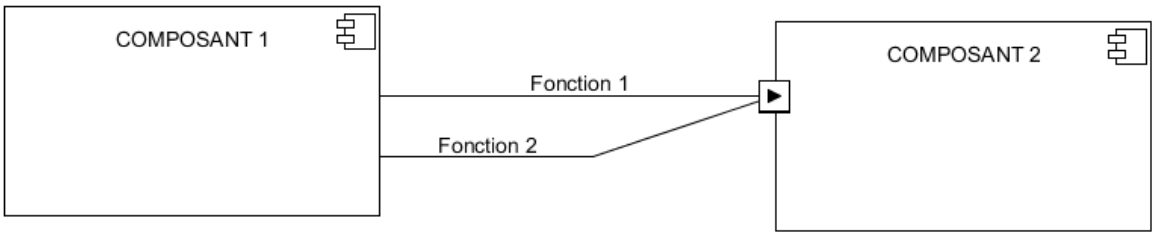
Cela signifie que le COMPOSANT 2 est un "serveur" puisqu'il doit se mettre en attente de réception de la "requête" Fonction 1.

Dans le cas, très courant, où le COMPOSANT 2 est sollicité par deux fonctions :

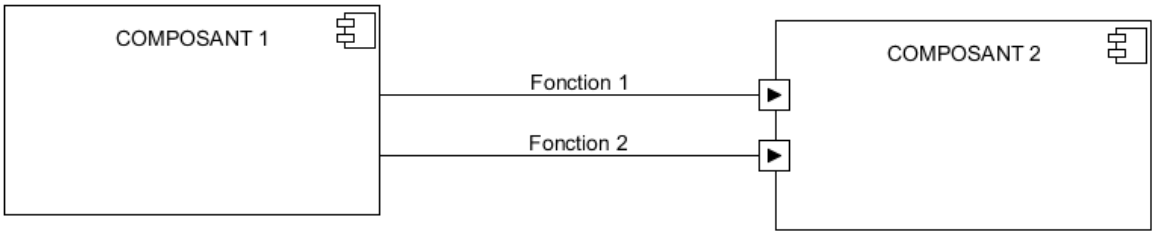


Cette description ne permet pas de savoir si le COMPOSANT 2 reçoit les "requêtes" à un seul endroit (1 seule classe Server) ou deux (2 classes Serveur).

On peut affiner cette description ainsi :

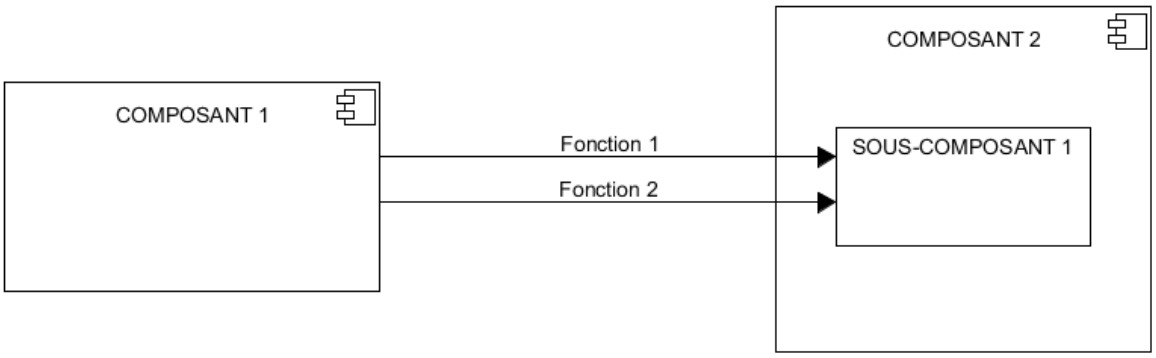


OU



On crée sur le COMPOSANT 2 ce que l'on appelle en architecture un "port" de communication.

On peut aussi faire apparaître la notion de sous-composants :

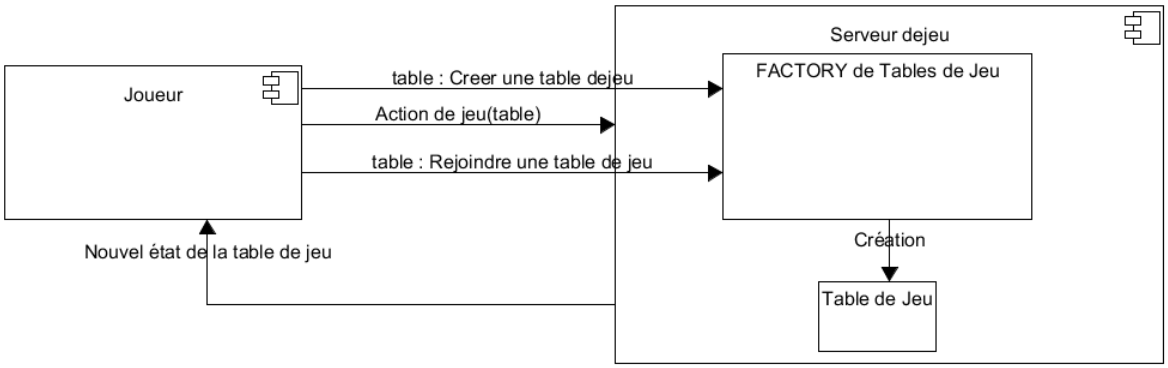


Ici, le SOUS COMPOSANT 2 traite les deux Fonction 1 et 2, il est donc le seul point d'entrée. Inutile ici de représenter le port.

Le sens des connecteurs est important. Chaque connecteur traduit un appel à un traitement, et ne traduit jamais un flux de communication.
L'orientation est donc la même pour pousser (push) des données à un composant ou pour lui tirer (pull) des données.
Les données poussées sont en paramètre d'entrée de la fonction.
Les données tirées sont en paramètre de sortie de la fonction.

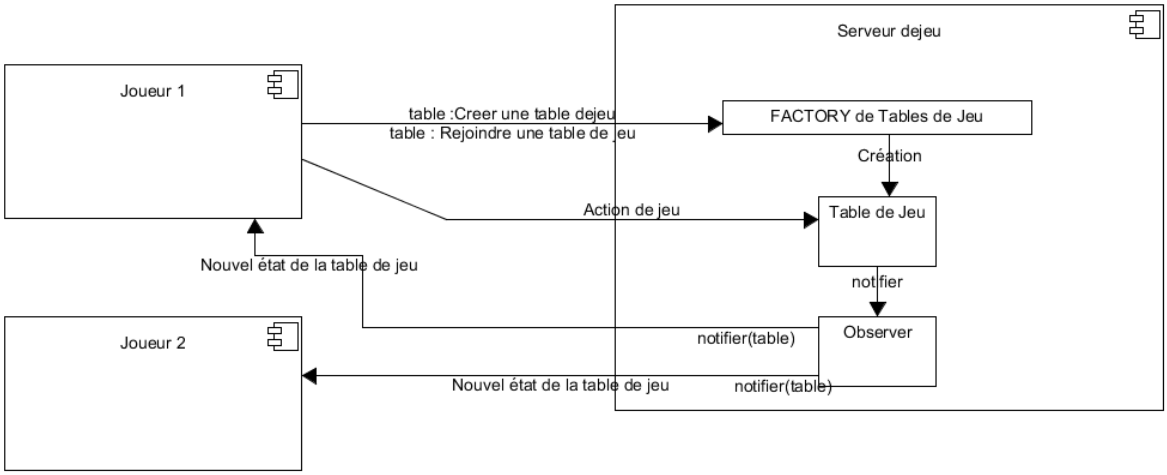
Prenons un exemple concret (jeu de poker en ligne) :

Une première représentation :



Commentaires en cours

Une deuxième représentation un peu plus détaillée :



Commentaires en cours

3.1.4 Valider son diagramme de communication

Pour valider son diagramme de communication, on « projette » dans le diagramme les fonctionnalités du SI décrits dans l'énoncé.

Une fonctionnalité se vérifie en suivant un chemin à travers les composants.

Chaque connecteur porte une ou plusieurs fonctionnalités.

3.1.5 Exemple

3.1.5.1 Le cahier des charges

Prenons un exemple concret : permettre à un utilisateur de faire une commande de produits, sur Internet, dans un catalogue. Il remplit un panier virtuel, passe sa commande. Sa commande est ensuite traitée pour préparer la livraison.

Si lors de la validation de son panier, un produit n'est pas disponible, il est prévenu et sa commande ne peut pas être validée.

Les commandes sont historisées dans le compte de l'utilisateur.

Les comptes, les commandes et les produits sont gérés en base de données.

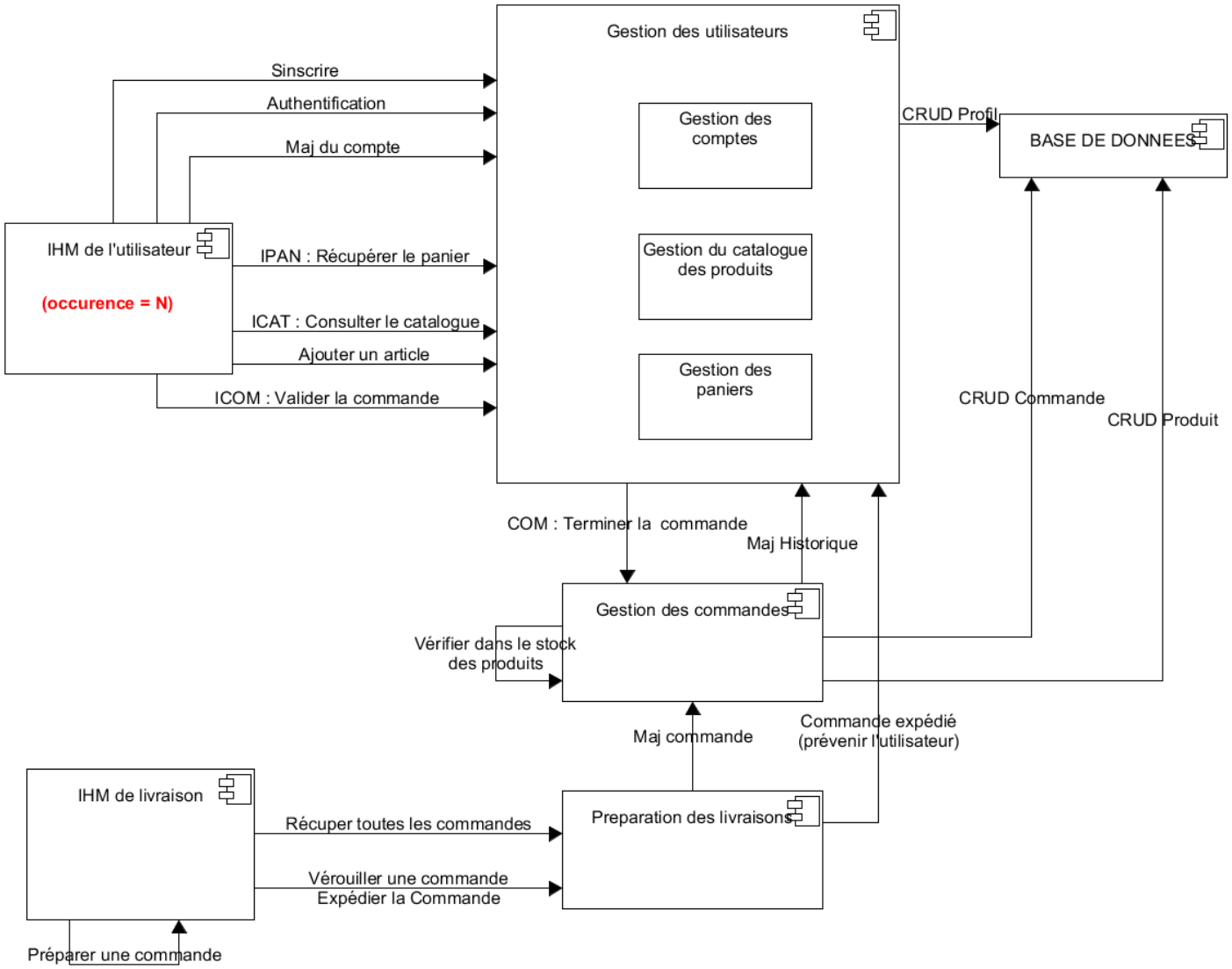
Un poste de livraison permet de préparer les commandes pour les livrer.

On a 5 composants logiciels :

- l' IHM d'un utilisateur
- l' IHM d'un poste de livraison
- le composant logiciel qui réalise les traitements de l'utilisateur qui fait sa commande
- le composant logiciel qui gère les commandes (au sens métier)
- le composant logiciel qui gère les livraisons (au sens métier)

3.1.5.2 Un simple diagramme de communication

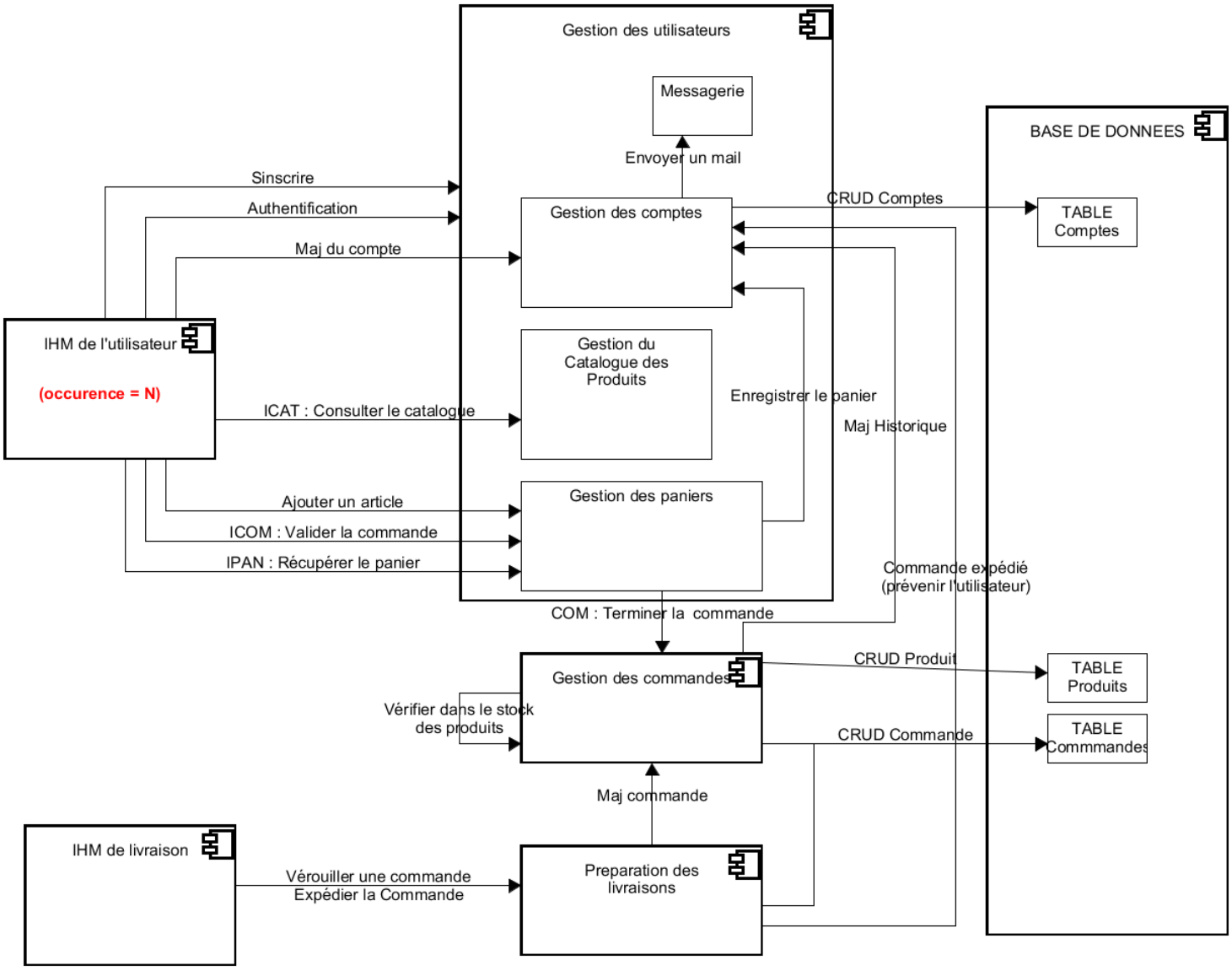
Une première représentation :



Commentaires en cours

Une deuxième représentation un peu plus détaillée dans le § suivant.

3.1.5.3 Un diagramme de communication plus complet



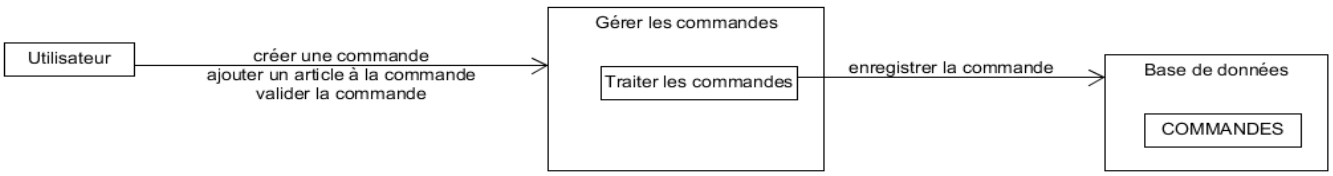
Commentaires en cours

4 Les données dans un Diagramme de Communication

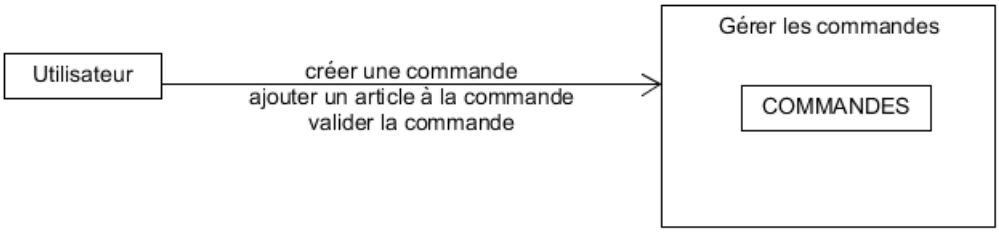
Prenons un cas fonctionnelle classique : je veux faire une commande d'articles.

Commentaires en cours

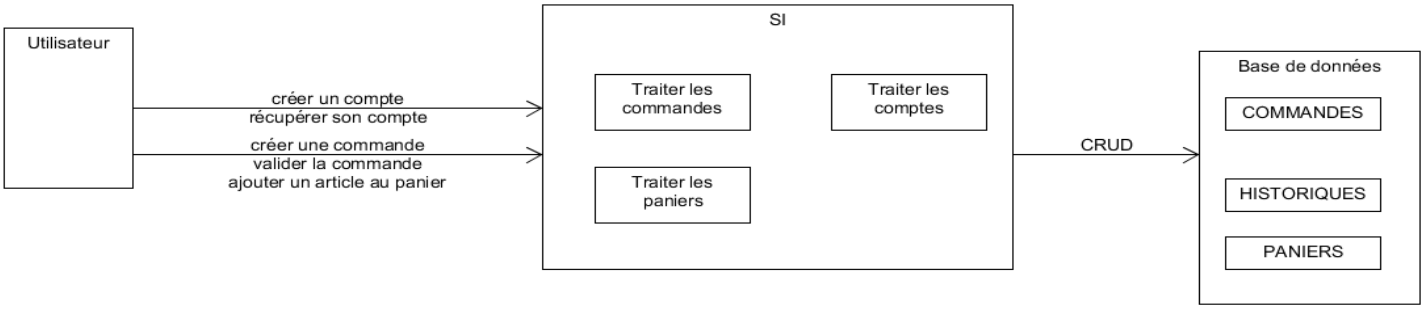
4.1 Stateless de commande



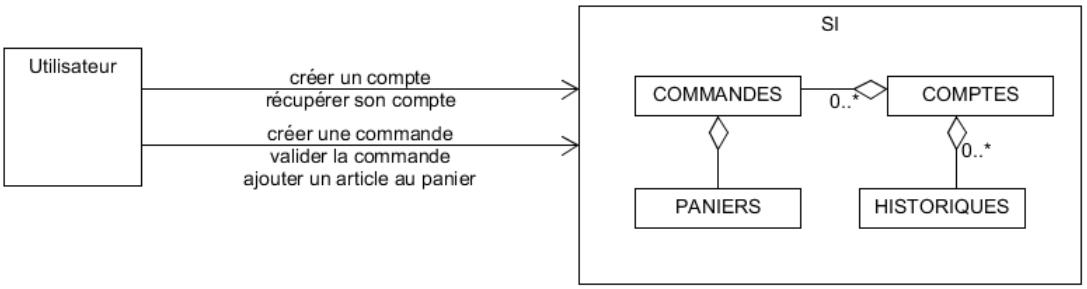
4.2 Statefull de commandes



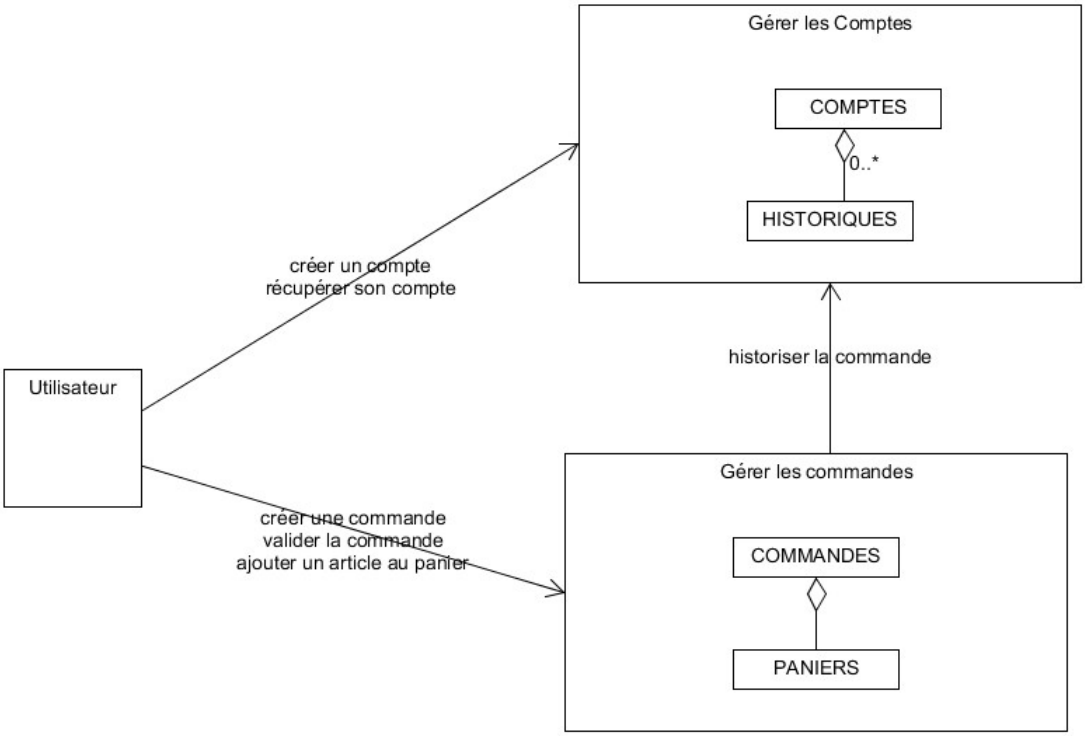
4.3 Stateless de commandes, comptes et paniers



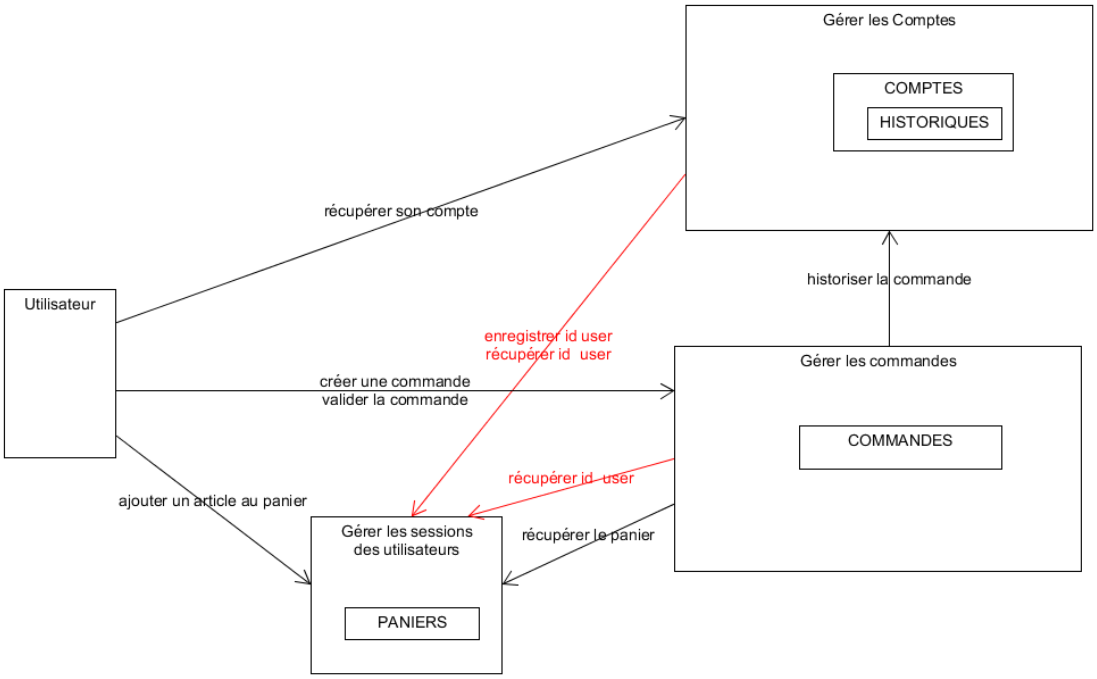
4.4 Statefull de commandes, comptes et paniers



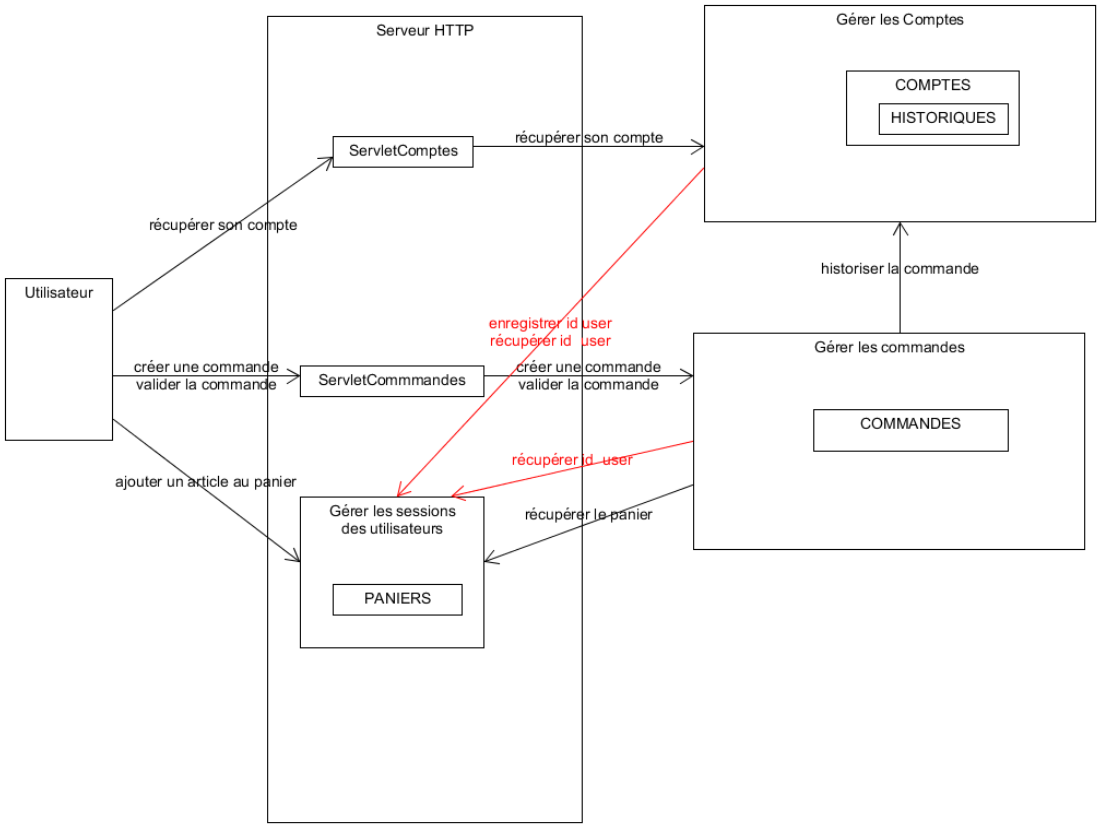
4.5 Panier de commande en Statefull



4.6 Panier de commande en session



4.7 Solution Internet



5 Choix du connecteur

Pour décrire les diagrammes de classes des COMPOSANTS LOGICIELS, nous devons faire un choix sur la technologie utilisée pour réaliser la communication entre les COMPOSANTS LOGICIELS.

Il existe plusieurs solutions en fonction des choix d'architecture :

- Communication Socket
- Communication JMS
- Communication RMI
- Communication Web Services :
  - SOAP
  - REST Full
- Architecture MOM avec JMS, RMI, ou Web Services
- Architecture MicroService avec REST
- ….

Quelque soit la technologie du connecteur utilisé, il est facile de représenter cela sous la forme d'un DP (Chapitre 5 §4).

Nous faisons le choix du protocole RMI pour les raisons suivantes :

- Simplicité de mise en œuvre
- Facile à décrire sous forme de diagramme de classe sans être obligé de faire des raccourcis

6 Exemples des corrections des examens

Voir sur le site.

7 Conclusion

L'objectif de ce cours, et le jour de l'examen, est d'écrire les diagrammes de classe des composants logiciels en utilisant les Design Patterns vus en cours.

Ainsi le diagramme de communication est une démarche de conception préliminaire qui doit vous permettre de savoir comment vous allez écrire vos classes principales et donc vos DP.

Pour cela, il est nécessaire de savoir COMMENT vous imaginez le fonctionnement des composants, comment ils vont communiquer entre eux, quel sont les composants qui stockent des données en mémoire, ...

Aussi, cela me permet de mieux comprendre vos diagrammes de classes. C'est pourquoi ce diagramme de communication est important, et est noté le jour de l'examen.

Il est IMPERATIF que le diagramme de communication soit en accord avec les diagrammes de classe.